

Amendments to the Claims

Please amend claims as follows.

1. (canceled)
2. (previously amended) A business-to-business application service provider with a software tool environment offered on a pay-per-use basis for system-on-a-chip designers to create unique intellectual property and providing an on-demand electronic design automation (EDA) computer program hosted on an Internet website, comprising the steps of:
 - generating an electronic circuit design;
 - partitioning said electronic circuit design into its constituent blocks and protocol design;
 - coding said constituent blocks and protocol design in hardware description language (HDL);
 - using high-level synthesis (HLS) for operation scheduling and resource allocation of said constituent blocks and protocol design;
 - technology-independent Boolean optimizing said constituent blocks after operation scheduling and resource allocation to produce an intermediate design;
 - technology-mapping said intermediate design to select particular devices for a hardware implementation of said electronic circuit design;
 - placing said particular devices at locations in a semiconductor chip; and

Atty. Dkt. No. MERO0001

routing a set of interconnections of said particular devices;
wherein, the step of technology-mapping comprises the sub-steps of:
partitioning an original circuit design into a set of corresponding logic
trees;

ordering said set of corresponding logic trees into an ordered linear list
such that each tree-T that drives another ordered tree precedes said other
ordered tree, and such that each ordered tree that drives said tree-T precedes
said tree-T;

sweeping forward in said ordered linear list while computing a set of
Pareto-optimal load/arrival curves for each of a plurality of net nodes that match
a technology-library element; and

sweeping backward in said ordered linear list while using said set of
Pareto-optimal load/arrival curves for each of said net nodes and a capacitive
load to select a best one of said technology-library elements with a shortest
signal arrival time;

wherein, only net nodes that correspond to a gate input are considered,
and any capacitive loads are predetermined.

3. (previously amended) A business-to-business application service
provider with a software tool environment offered on a pay-per-use basis for
system-on-a-chip designers to create unique intellectual property and providing
an on-demand electronic design automation (EDA) computer program hosted on
an Internet website, comprising the steps of:

partitioning an original circuit design into a set of corresponding logic trees

replacing each said logic tree with an equivalent simplified tree having no interior nodes;

analyzing each path from a tree leaf to its root in said original circuit;

computing a propagation delay for each said path; and

annotating computed delays onto corresponding arcs of said simplified trees.

4. (original) The business-to-business application service provider of claim 3, wherein said electronic design automation (EDA) computer program further provides for:

annotating any dependency of a propagation delay of said original circuit design on a slew rate of an input signal onto a corresponding leaf of said simplified tree.

5. (original) The business-to-business application service provider of claim 3, wherein said electronic design automation (EDA) computer program further provides for:

copying any capacitive load values from any leaves of said logic tree to corresponding leaves of said simplified tree.

6. (original) The business-to-business application service provider of claim 3, wherein said electronic design automation (EDA) computer program further provides for:

copying a load/delay response curve of an output gate at an apex of said logic tree, to a root of said simplified tree.

7. (original) The business-to-business application service provider of claim 3, wherein said electronic design automation (EDA) computer program further provides for:

collapsing an entire delay calculation into a simple edge-weighted longest-path traversal within an interior of an abstract timing model of a resource.

8. (original) The business-to-business application service provider of claim 3, wherein said electronic design automation (EDA) computer program further provides for:

calculating timing delays for an electronic design with a combination of complex-model trees that interface circuit boundaries and simple-model trees that are internal and do not interface with circuit boundaries.

9. (original) The business-to-business application service provider of claim 3, wherein said electronic design automation (EDA) computer program further provides a technology selection process comprising the steps of:

partitioning an original circuit design into a set of corresponding logic trees;

ordering said set of corresponding logic trees into an ordered linear list such that each tree-T that drives another ordered tree precedes said other

Atty. Dkt. No. MERO0001

ordered tree, and such that each ordered tree that drives said tree-T precedes said tree-T;

sweeping forward in said ordered linear list while computing a set of Pareto-optimal load/arrival curves for each of a plurality of net nodes that match a technology-library element; and

sweeping backward in said ordered linear list while using said set of Pareto-optimal load/arrival curves for each of said net nodes and a capacitive load to select a best one of said technology-library elements with a shortest signal arrival time;

wherein, only net nodes that correspond to a gate input are considered, and any capacitive loads are predetermined.

10. (original) The business-to-business application service provider of claim 3, wherein said electronic design automation (EDA) computer program further provides for:

generating an electronic circuit design;

partitioning said electronic circuit design into its constituent blocks and protocol design;

coding said constituent blocks and protocol design in hardware description language (HDL);

using high-level synthesis (HLS) for operation scheduling and resource allocation of said constituent blocks and protocol design;

technology-independent Boolean optimizing said constituent blocks after operation scheduling and resource allocation to produce an intermediate design;

Atty. Dkt. No. MERO0001

technology-mapping said intermediate design to select particular devices for a hardware implementation of said electronic circuit design;

placing said particular devices at locations in a semiconductor chip; and

routing a set of interconnections of said particular devices;

wherein, the step of technology-mapping comprises the sub-steps of:

partitioning an original circuit design into a set of corresponding logic trees;

ordering said set of corresponding logic trees into an ordered linear list such that each tree-T that drives another ordered tree precedes said other ordered tree, and such that each ordered tree that drives said tree-T precedes said tree-T;

sweeping forward in said ordered linear list while computing a set of Pareto-optimal load/arrival curves for each of a plurality of net nodes that match a technology-library element; and

sweeping backward in said ordered linear list while using said set of Pareto-optimal load/arrival curves for each of said net nodes and a capacitive load to select a best one of said technology-library elements with a shortest signal arrival time;

wherein, only net nodes that correspond to a gate input are considered, and any capacitive loads are predetermined.

11. (original) The business-to-business application service provider of claim 10, wherein:

Atty. Dkt. No. MERO0001

the step of using high-level synthesis is such that a timing analysis is applied each time an individual operation is scheduled, and may called many times to get a single operation scheduled.

12. (original) The business-to-business application service provider of claim 10, wherein:

the step of technology mapping maps abstract Boolean gates of said electronic circuit design to standard cells from a technology library.

13. (original) The business-to-business application service provider of claim 3, wherein said electronic design automation (EDA) computer program further provides for transforming a hardware-description language text representing a sequential program into a control-flow graph for later operation scheduling and technology allocation by:

reducing a hardware-description language text representing a sequential program into a control-flow graph;

constructing a one-hot-bit finite state machine from said control-flow graph; and

predicting an operational timing of said one-hot-bit finite state machine before operation scheduling in an electronic design automation system;

wherein, a cycle-by-cycle timing congruence is maintained between said hardware-description language text and a final synthesized design.

Atty. Dkt. No. MERO0001

14. (original) The business-to-business application service provider of claim 13, wherein:

the reducing comprises a step-by-step reduction of a parse tree; wherein, particular parse tree structures are recognized and corresponding subgraphs are constructed.

15. (original) The business-to-business application service provider of claim 14, wherein: the reducing begins with the construction of a simple graph having a reset node and a join node with a trivial self-loop; wherein an "always" construct is implemented.

16. (original) The business-to-business application service provider of claim 15, wherein:

the reducing continues by transforming said simple graph into a more elaborate control-flow graph by applying a procedure to any parse-tree statements annotated onto any arcs;

wherein an arc that has a statement is removed and replaced with at least two new arcs and at least one new node.

17. (original) The business-to-business application service provider of claim 16, wherein: the reducing continues by applying said procedure recursively to all said parse-tree statements annotated onto all said arcs;

wherein all decomposable statements are processed.

18. (original) The business-to-business application service provider of claim 17, wherein:

the reducing continues by saving a name of any labeled block in a table that maps such name to a node "end";

wherein said node "end" provides a destination for any Verilog "disable" statements.

19. (original) The business-to-business application service provider of claim 17, wherein:

the reducing continues by introducing an iteration-counter variable in a case of a "repeat" loop;

wherein said iteration-counter variable is initialized before entering said loop and is incremented each time said loop rolls around.

20. (original) The business-to-business application service provider of claim 17, wherein:

the reducing continues by attaching a condition to each of two out-arcs of a new node "iter" for a "repeat" loop, a "while" loop, or a "for" loop.

21. (original) The business-to-business application service provider of claim 17, wherein:

the reducing continues by removing any arcs and nodes that are unreachable by a forward transversal from a "reset" node.

Atty. Dkt. No. MERO0001

22. (original) The business-to-business application service provider of claim 17, wherein:

the reducing continues by collapsing together any sets of arcs resulting from branches containing no further graph structure and re-annotating any conditional parse trees onto said control-flow graph.

23. (original) The business-to-business application service provider of claim 17, wherein:

the reducing continues by detecting if any conditionals have no effect on said control-flow graph other than a creation of a surplus branch, and if so not applying a reduction and annotating said conditionals as they are.

24. (original) The business-to-business application service provider of claim 17, wherein:

the reducing continues by removing any dead branches whose conditions can never be true.

25. (original) The business-to-business application service provider of claim 17, wherein:

the reducing continues by merging the in-arcs and out-arcs of simple nodes with one in-arc and one out-arc that are not marked as states.

Atty. Dkt. No. MERO0001

26. (original) The business-to-business application service provider of claim 17, further including:

pruning of said control-flow graph after the step of reducing to accommodate a Verilog "disable" statement in said hardware-description language text.

27. (original) The business-to-business application service provider of claim 17, further including:

the pruning of said control-flow graph after the step of reducing to accommodate a "goto" statement in said hardware-description language text.

28. (original) The business-to-business application service provider of claim 13, wherein:

the step of constructing said one-hot-bit finite state machine includes mapping each state node of said control-flow graph to a corresponding single-state flip-flop;

wherein, a respective state is indicated when any flip-flop output is true.

29. (currently amended) The business-to-business application service provider of claim 16, wherein:

the step of constructing said one-hot-bit finite state machine continues by constructing a table MAP;

wherein[,] any arcs of said control-flow graph are mapped to a corresponding output port of said finite state machine[.]; wherein MAP and PAM

Atty. Dkt. No. MERO0001

are two tables or functions and MAP maps an edge of the control-flow graph to a pin of the circuit and PAM maps a pin of the circuit to an edge of the control-flow graph. PAM being the inverse of MAP and MAP is the inverse of PAM.

30. (currently amended) The business-to-business application service provider of claim 29, wherein:

the step of constructing said one-hot-bit finite state machine continues by building a circuit which is driven by a set of primary inputs and state flip-flops, and which drives a MAP(C)[,]; wherein MAP and PAM are two tables or functions and

MAP maps an edge of the control-flow graph to a pin of the circuit and PAM maps a pin of the circuit to an edge of the control-flow graph. PAM being the inverse of MAP and MAP is the inverse of PAM.

31. (currently amended) The business-to-business application service provider of claim 30, wherein:

the step of constructing said one-hot-bit finite state machine includes using a procedure approximated by,

Procedure cct(c)

T = PAM(c);

N = { the node at the feather end of C }

If (N is a state node) {

Connect T to the Q pin of FLOP(N).

} else if (N is a join node with K in-arcs) {

Atty. Dkt. No. MERO0001

```
G = a new k-input OR gate
Connect the output of G to T.
for (each in-arc A of N) {
    Let P = MAP(A).
    If (P is null) {
        construct a new output pin named P.
        set MAP(A) = P.
        call cct(A).}
    connect P to an unconnected input pin of G.}
} else if (N is a fork node) {
    Construct a new 2-input AND gate G.
    Connect the output of G to T.
    Let A be the in-arc of N.
    Let P = MAP(A).
    If (P is null) {
        Construct a new output pin named P.
        Set map(A) = P
        Call cct(A).
    }
    Connect one input of G to P
    Make the other input of G a primary status input
    Corresponding to the branch condition
    That is annotated onto A.
} else if (N is the reset node) {
```

Atty. Dkt. No. MERO0001

Connect P to the reset input.

}else{

Let A be the in-arc of N.

Let P = MAP(A).

If (P is null) {

Construct a new output pin named P.

Set map(A) = P

Call cct(A).

}

Connect P to T.

}

end;

wherein MAP and PAM are two tables or functions and

MAP maps an edge of the control-flow graph to a pin of the circuit and
PAM maps a pin of the circuit to an edge of the control-flow graph, PAM being
the inverse of MAP and MAP is the inverse of PAM; and

wherein the procedure CCT takes as its sole argument an arc C of the
control-flow graph and performs a recursive backward traversal of the control
flow graph that builds the desired one-hot FSM as it traverses the control-flow
graph, using only local information about the graph and the tables MAP, PAM,
and FLOP.